


Leszek Rutkowski Marcin Korytkowski  
Rafał Scherer Ryszard Tadeusiewicz  
Lotfi A. Zadeh Jacek M. Zurada (Eds.)

LNAI 7268

# Artificial Intelligence and Soft Computing

11th International Conference, ICAISC 2012  
Zakopane, Poland, April/May 2012  
Proceedings, Part II

 Part II

 Springer

## Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada  
Jörg Siekmann, University of Saarland, Saarbrücken, Germany  
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

## Volume Editors

Leszek Rutkowski  
Marcin Korytkowski  
Rafał Scherer  
Częstochowa University of Technology, Poland  
E-mail: lrutko@kik.pcz.czest.pl.  
{marcin.korytkowski, rafal.scherer}@kik.pcz.pl

Ryszard Tadeusiewicz  
AGH University of Science and Technology, Kraków, Poland  
E-mail: rtad@agh.edu.pl

Lotfi A. Zadeh  
University of California, Berkeley, CA, USA  
E-mail: zadeh@cs.berkeley.edu

Jacek M. Zurada  
University of Louisville, KY, USA  
E-mail: jacek.zurada@louisville.edu

ISSN 0302-9743 e-ISSN 1611-3349  
ISBN 978-3-642-29349-8 e-ISBN 978-3-642-29350-4  
DOI 10.1007/978-3-642-29350-4  
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012934672

CR Subject Classification (1998): I.2, H.3, F.1, I.4, H.4, I.5

LNCS Sublibrary: SL 7 – Artificial Intelligence

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

*Typesetting:* Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Fast Parallel Cost Function Calculation for the Flow Shop Scheduling Problem\*

Wojciech Bożejko<sup>1</sup>, Mariusz Uchroński<sup>1,2</sup>, and Mieczysław Wodecki<sup>3</sup>

<sup>1</sup> Institute of Computer Engineering, Control and Robotics,  
Wrocław University of Technology,  
Janiszewskiego 11-17, 50-372 Wrocław, Poland  
wojciech.bozejko@pwr.wroc.pl

<sup>2</sup> Wrocław Centre of Networking and Supercomputing,  
Wyb. Wyspańskiego 27, 50-370 Wrocław, Poland  
mariusz.uchronski@pwr.wroc.pl

<sup>3</sup> Institute of Computer Science, University of Wrocław,  
Joliot-Curie 15, 50-383 Wrocław, Poland  
mwd@ii.uni.wroc.pl

**Abstract.** In this paper we are proposing a methodology of the fast determination of the objective function for the flow shop scheduling problem in a parallel computing environment. Parallel Random Access Machine (PRAM) model is applied for the theoretical analysis of algorithm's efficiency. The presented method needs a fine-grained parallelization, therefore the proposed approach is especially devoted to parallel computing systems with fast shared memory, such as GPUs.

## 1 Introduction

We can see the process of jobs flowing through machines (processors) in many practical problems of scheduling: in computer systems as well as in production systems. Thus, the flow shop scheduling problem represents a wide class of possible applications, depending on the cost function definition. For each of them, a corresponding discrete model has to be constructed and analyzed. Some of them (e.g. with the makespan criterion and with total weighted tardiness cost function) have got a special elimination-criteria (so-called *block properties*) which speed up the calculation significantly, especially in the multithread computing environment.

### 1.1 Formulation of the Problem

The problem has been introduced as follows. There are  $n$  jobs from a set  $\mathcal{J} = \{1, 2, \dots, n\}$  to be processed in a production system having  $m$  machines, indexed by  $1, 2, \dots, m$ , organized in the line (sequential structure). A single job reflects one final product (or sub product) manufacturing. Each job is performed in  $m$

\* The work was supported by MNiSW Poland, within the grant No. N N514 23223.

subsequ  
by a ma  
operatio  
processi  
job cann  
each job

The s  
 $\pi = (\pi(i$   
find the

where  $C$   
permuta  
job  $j$  on  
by using

$i = 1, 2,$   
 $C_{0\pi(j)} =$

Comput

The for  
that the  
into pa  
comput:

Gare  
rion  $C_m$   
local se  
Taillard  
[9]. Bož  
method  
also pre  
pera [5]  
proben  
and We  
for the  
conside  
allelizat  
for the

subsequent stages, in a way common to all the tasks. The stage  $i$  is performed by a machine  $i$ ,  $i = 1, 2, \dots, m$ . Each job  $j \in \mathcal{J}$  is split into a sequence of  $m$  operations  $O_{1j}, O_{2j}, \dots, O_{mj}$  performed on machines. The operation  $O_{ij}$  reflects processing of job  $j$  on machine  $i$  with processing time  $p_{ij} > 0$ . Once started the job cannot be interrupted. Each machine can execute at most one job at a time, each job can be processed on at most one machine at a time.

The sequence of loading jobs into a system is represented by a permutation  $\pi = (\pi(1), \dots, \pi(n))$  of elements of the set  $\mathcal{J}$ . The optimization problem is to find the optimal sequence  $\pi^*$  so that

$$C_{\max}(\pi^*) = \min_{\pi \in \Phi_n} C_{\max}(\pi) \tag{1}$$

where  $C_{\max}(\pi)$  is the makespan for a permutation  $\pi$  and  $\Phi_n$  is the set of all permutations of elements of the set  $\mathcal{J}$ . Denoted by  $C_{ij}$  the completion time of job  $j$  on machine  $i$  we obtain  $C_{\max}(\pi) = C_{m, \pi(n)}$ . The values  $C_{ij}$  can be found by using either the recursive formula

$$C_{i\pi(j)} = \max\{C_{i-1, \pi(j)}, C_{i, \pi(j-1)}\} + p_{i\pi(j)}. \tag{2}$$

$i = 1, 2, \dots, m$ ,  $j = 1, 2, \dots, n$ , with initial conditions  $C_{j\pi(0)} = 0$ ,  $i = 1, 2, \dots, m$ ,  $C_{0\pi(j)} = 0$ ,  $j = 1, 2, \dots, n$ , or a non-recursive one

$$C_{i\pi(j)} = \max_{1=j_0 \leq j_1 \leq \dots \leq j_i=j} \sum_{s=1}^i \sum_{k=j_{s-1}}^{j_s} p_{s\pi(k)}. \tag{3}$$

Computational complexity of (2) is  $O(mn)$ , whereas for (3) it is

$$O\left(\binom{j+i-2}{i-1}(j+i-1)\right) = O\left(\frac{(n+m)^{n-1}}{(n-1)!}\right). \tag{4}$$

The former formula has been commonly used in practice. It should be noticed that the problem of transforming sequential algorithm for scheduling problems into parallel one is nontrivial because of the strongly sequential character of computations carried out using (2) and other known scheduling algorithms.

Garey et al. [8] showed that the flow shop problem with makespan criterion  $C_{\max}$  is strongly NP-hard for  $m \geq 3$  machines. Various serial and parallel local search methods are available. Tabu search algorithms were proposed by Taillard [12], Reeves [11], Nowicki and Smutnicki [10], Grabowski and Wodecki [9]. Bożejko and Wodecki [3] applied this method in the parallel path-relinking method used to solve the flow shop scheduling problem. Bożejko and Wodecki also proposed a parallel scatter search [4] for this problem. Bożejko and Pempera [5] presented a parallel tabu search algorithm for the permutation flow shop problem of minimizing the criterion of the sum of job completion times. Bożejko and Wodecki [6] proposed applying multi-moves in parallel genetic algorithm for the flow shop problem. The theoretical properties of these multi-moves were considered by Bożejko and Wodecki in the paper [7]. A survey of single-walk parallelization methods of the cost function calculation and neighborhood searching for the flow shop problem can be found in Bożejko [1].

1.2 Models

The values  $C_{ij}$  from equations (2) and (3) can also be determined by means of a graph model of the flow shop problem. For a given sequence of job execution  $\pi \in \Phi_n$  we create a graph  $G(\pi) = (M \times \mathcal{J}, F^0 \cup F^s)$ , where  $M = \{1, 2, \dots, m\}$ ,  $\mathcal{J} = \{1, 2, \dots, n\}$ .

$$F^0 = \bigcup_{s=1}^{m-1} \bigcup_{t=1}^n \{(s, t), (s + 1, t)\} \tag{5}$$

is a set of technological arcs (vertical) and

$$F^s = \bigcup_{s=1}^m \bigcup_{t=1}^{n-1} \{(s, t), (s, t + 1)\} \tag{6}$$

is a set of sequencing arcs (horizontal).

Arcs of the graph  $G(\pi)$  have no weights but each vertex  $(s, t)$  has a weight  $p_{s\pi(t)}$ . A time  $C_{ij}$  of finishing a job  $\pi(j)$ ,  $j = 1, 2, \dots, n$  on machine  $i$ ,  $i = 1, 2, \dots, m$  equals the length of the longest path from vertex  $(1, 1)$  to vertex  $(i, j)$  including the weight of the last one. A sample 'mesh' graph  $G(\pi)$  is shown in Fig. 1. The mesh is always the same, vertices weights depend on the  $\pi$ . For the flow shop problem with  $C_{\max}$  cost function the value of the criterion function for a fixed sequence  $\pi$  equals the length of the critical path in the graph  $G(\pi)$ . For the flow shop problem with the  $C_{\text{sum}}$  criterion the value of the criterion function is the sum of lengths of the longest paths which begin from vertex  $(1, 1)$  and ends on vertices  $(m, 1), (m, 2), \dots, (m, n)$ .

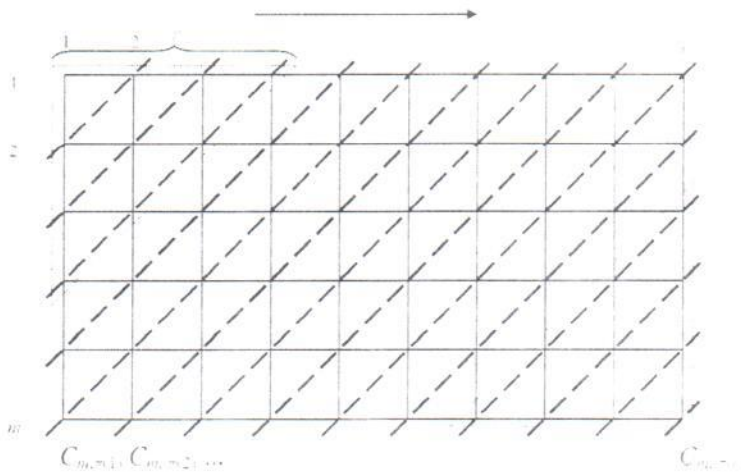


Fig. 1. Scheme of a fast parallel cost function calculation in the flow shop

The graph  $G(\pi)$  is also strongly connected with formulas (2) and (3) of completion times  $C_{ij}$  calculation. By using formula (2), it is enough to generate consecutive vertices, as dashed lines show (see Fig. 1) taking in the vertex  $(i, j)$ ,

connected upper on path in t the longe izontal st

2 Pa

The rec  $C_{ij}$  on p time  $T_s$

There 1, 2, ..., theoreti generali (2) have  $k = 1, 2$  in an or most  $m$  ground by dash  $C_{m,n}$  ] be perf with th

As we many s iterat speedt

In this

Here in wh proce Th  $C_{1,1}$  a "tri

connected with the  $C_{ij}$ , a greater value from the left vertex,  $C_{i,j-1}$ , and from the upper one,  $C_{i-1,j}$ , and adding  $p_{ij}$  to it. Such a procedure generates the longest path in the graph  $G(\pi)$  in time  $O(nm)$ . Formula (3) can also be presented as the longest path generation algorithm but its conception is based on the all horizontal sub-paths generation and its computational complexity is exponential.

## 2 Parallel Cost Function Determination

The recurrent formula (2) is applied to determination times of jobs completion  $C_{ij}$  on particular machines. With the use of a single processor, the calculations time  $T_s$  of the  $C_{i\pi(j)}$  value (according to (2)) is  $O(nm)$ .

There is a method of times of jobs finishing determination  $C_{i\pi(j)}$  ( $i = 1, 2, \dots, m$ ,  $j = 1, 2, \dots, n$ ) on the  $p$  ( $p < m$ ) -processors CREW PRAM (a theoretical parallel calculations model) in the paper [2]. Without the loss of generality let us assume that  $\pi = (1, 2, \dots, n)$ . Calculations of  $C_{i,j}$  by using (2) have been clustered. Cluster  $k$  contains values  $C_{ij}$  such that  $i + j - 1 = k$ ,  $k = 1, 2, \dots, n + m - 1$  and requires at most  $m$  processors. Clusters are processed in an order  $k = 1, 2, \dots, n + m - 1$ . The cluster  $k$  is processed in parallel on at most  $m$  processors. The sequence of calculations is shown in Fig. 2 on the background of a grid graph commonly used for the flow shop problem. Values linked by dashed lines constitute a single cluster. The value of  $C_{max}$  criterion is simple  $C_{m,n}$ . To calculate  $C_{sum} = \sum_{j=1}^n C_{m,j}$  we need to add  $n$  values  $C_{m,j}$ , which can be performed sequentially in  $n$  iterations or in parallel by using  $m$  processors with the complexity  $O(n/m + \log m)$ . In this case, the calculation time is

$$T_p = \frac{nm}{p} + n + m - 1. \tag{7}$$

As we can see, calculations are made in groups of  $p$  processors and there are many situations, in which some part of processors from a group has an idle time (iterations 1, 2, 3, 6, 8, 10, 12, 14, 16, 18, 20, 21, 22 in Fig. 2). Therefore, the speedup equals

$$s_p = \frac{T_s}{T_p} = \frac{nm}{\frac{nm}{p} + n + m - 1}. \tag{8}$$

In this case a limiting value of the speedup is

$$\lim_{n \rightarrow \infty} s_p = \lim_{n \rightarrow \infty} \frac{T_s}{T_p} = \lim_{n \rightarrow \infty} \frac{nm}{\frac{nm}{p} + n + m - 1} = \frac{mp}{m + p} = \frac{p}{1 + \frac{p}{m}}. \tag{9}$$

Here we are proposing the new method of calculation of jobs finishing times, in which we fully take advantage of multiprocessor environment by reducing processors idle times. The idea is shown in Fig. 3.

Therefore, in the beginning, the following elements are determined:  $C_{1,\pi(1)}, C_{1,\pi(2)}, C_{2,\pi(1)}, C_{1,\pi(3)}, C_{2,\pi(2)}, C_{3,\pi(1)}, \dots, C_{p,\pi(p)}$ ,  $p < m$ . These elements create a "triangle" with the width  $p$  in the Fig. 1, and their number is  $\frac{p(p+1)}{2}$ . The time

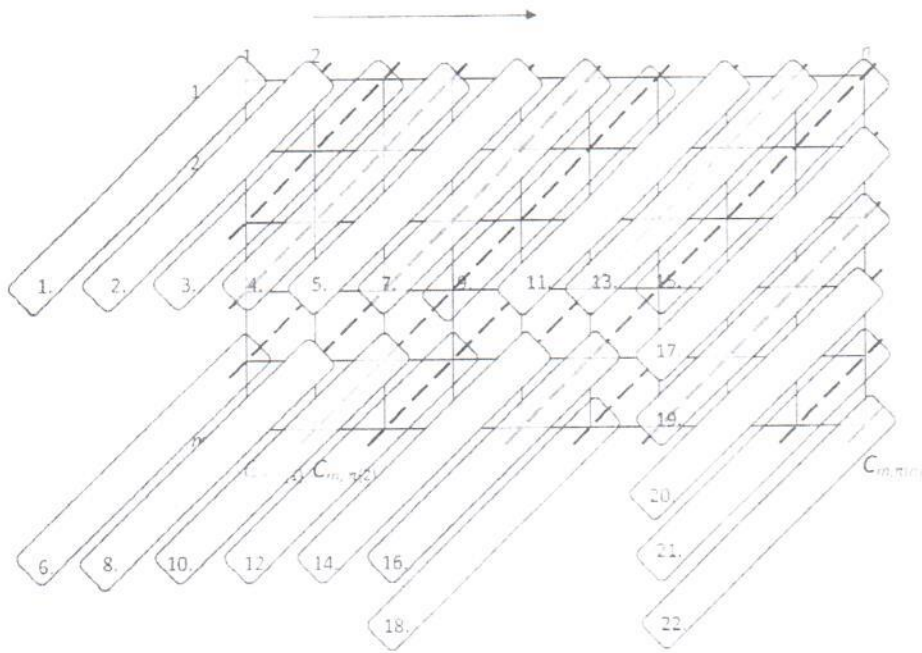


Fig. 2. Scheme of a classic parallel cost function calculation in the flow shop

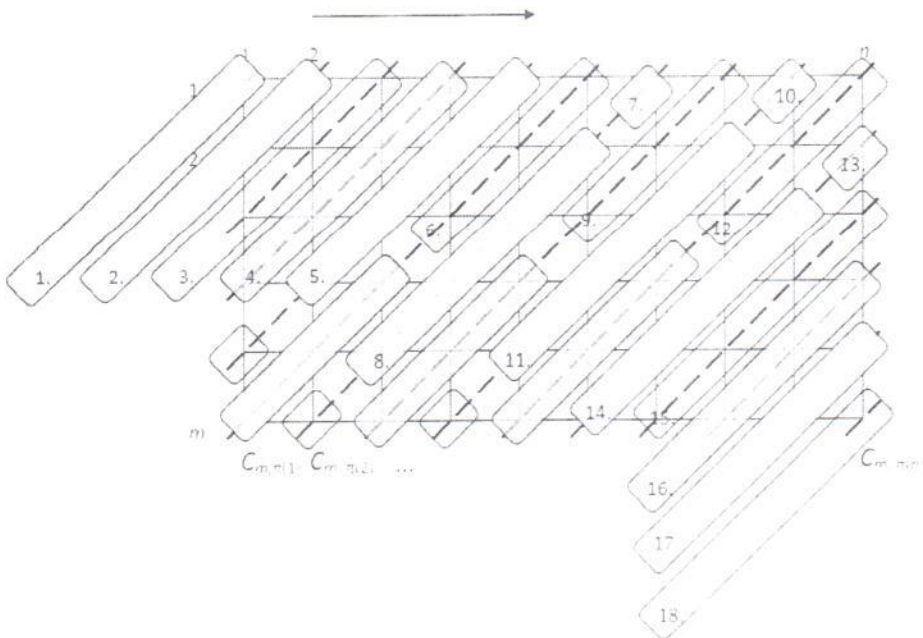


Fig. 3. Scheme of a fast parallel cost function calculation in the flow shop

of its  
idle t  
symm  
 $i = 1$   
elem  
betwe  
botto  
the re  
 $C_{1, \pi(i)}$   
Th  
the s  
There

$T_p^o$   
and c

Howe

Thus,  
taken

### 3 I

The p  
probl  
Tesla  
GPU  
ing ur  
 $m - 1$   
taine  
using  
value  
speed  
putati

of its calculation (on the  $p$ -processors CREW PRAM)) is  $p$  (but there are some idle times, as in iterations 1.2.3 in the Fig. 3). It is obvious, that the number of the symmetric lower right "triangle" is also  $\frac{p(p+1)}{2}$ . Because the number of all  $C_{i,\pi(j)}$ ,  $i = 1, 2, \dots, n$ ,  $j = 1, 2, \dots, m$  is  $nm$ , so there is  $nm - 2\frac{p(p+1)}{2} = nm - p(p+1)$  elements out of both "triangles". We do not generate idle times inside the region between "triangles" because if a group of processors calculates elements on the bottom of the graph, including the lowest element  $C_{m,\pi(k)}$ , for some  $k$ , then the rest of the processors, which normally have an idle time, calculates elements  $C_{1,\pi(k+m-1)}$ ,  $C_{2,\pi(k+m-3)}$ , etc.

The time of calculations of the region between "triangles", in accordance with the scheme presented in Fig. 3, is  $\frac{nm-p(p+1)}{p}$  on  $p$ -processors CREW PRAM. Therefore, the total calculations time, including "triangles", is

$$T_p^\circ = \frac{p(p+1)}{2} + \frac{nm-p(p+1)}{p} + \frac{p(p+1)}{2} = p(p+1) + \frac{nm-p(p+1)}{p} \quad (10)$$

and consequently the speedup equals

$$s_p^\circ = \frac{T_s}{T_p^\circ} = \frac{nm}{p(p+1) + \frac{nm-p(p+1)}{p}} \quad (11)$$

However, the limiting value of this speedup is

$$\lim_{n \rightarrow \infty} s_p^\circ = \lim_{n \rightarrow \infty} \frac{T_s}{T_p^\circ} = \lim_{n \rightarrow \infty} \frac{nm}{p(p+1) + \frac{nm-p(p+1)}{p}} = \quad (12)$$

$$= \lim_{n \rightarrow \infty} \frac{nm}{p^2(p+1) + nm - p(p+1)} = p. \quad (13)$$

Thus, we have obtained a much better limiting speedup comparing to the method taken from the paper [2].

### 3 Experimental Results

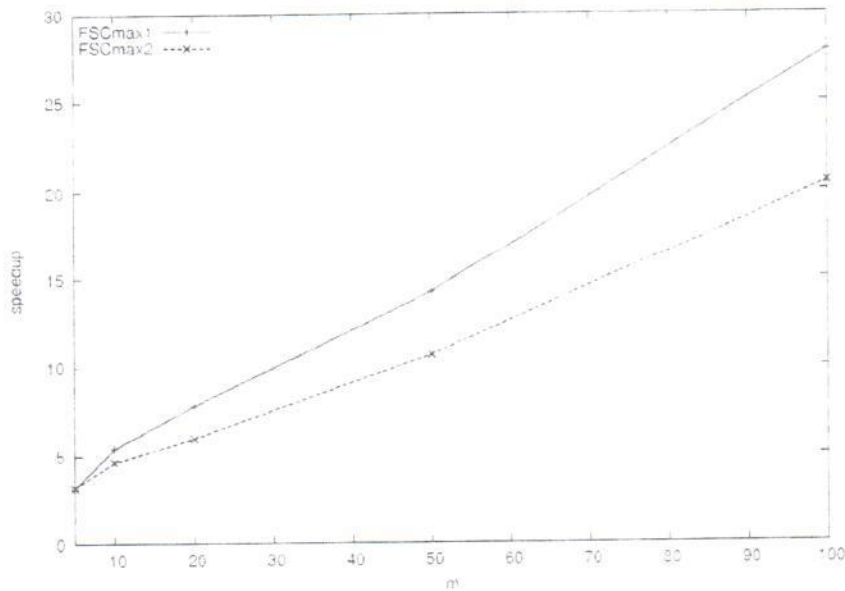
The parallel algorithm of calculating goal function in permutation flow shop problem was coded in C(CUDA) for GPU, ran on the 1792-processors nVidia Tesla S2050 GPU and tested on the benchmark problems of Taillard [12]. The GPU was installed on the server based on Intel Core i7 3.33GHz processor working under 64-bit Ubuntu 10.04 operating system. The algorithm considered uses  $m - 1$  GPU processors for calculating a makespan. Its sequential version is obtained by assigning  $p = 1$  and is also executed on GPU. The sequential algorithm, using one GPU processor, was coded with the aim of determining the speedup value of the parallel algorithm. Table 1 shows computational times for the sequential and parallel algorithm as well as a speedup. The value of a relative speedup  $s$  can be found by the following expression  $s = \frac{t_s}{t_p}$ , where  $t_s$  - the computational time of sequential algorithm executed on the single GPU processor,



$t_p$  - the computational time of parallel algorithm executed on  $p$  GPU processors. As we can notice the highest average speedup values were obtained for the problem instances with bigger number of jobs  $n$  and bigger number of machines  $m$ , because in these cases the influence of "triangles" regions (described in the Section 2) on the total parallel computations is the lowest, from the theoretical point of view.

**Table 1.** Experimental results for Taillard's instances

$n \times m$	$p$	$t_p [ms]$	$t_s [ms]$	speedup $s$
$20 \times 5$	4	0.0237	0.0665	2.7998
$20 \times 10$	9	0.0293	0.1461	4.9792
$20 \times 20$	19	0.0463	0.3694	7.9685
$50 \times 5$	4	0.0539	0.1583	2.9336
$50 \times 10$	9	0.0676	0.3356	4.9642
$50 \times 20$	19	0.1039	0.7460	7.1753
$100 \times 5$	4	0.1033	0.3157	3.0548
$100 \times 10$	9	0.1243	0.6460	5.1939
$100 \times 20$	19	0.1844	1.3762	7.4627
$200 \times 10$	9	0.2387	1.2717	5.3267
$200 \times 20$	19	0.3445	2.6378	7.6555
$500 \times 20$	19	0.8248	6.3980	7.7567



**Fig. 4.** Speedup comparison

TI  
algor  
show  
speed  
were  
a dif  
speed  
rithm

4

The  
meta  
goal  
of th  
funct  
time  
were

Ref

1. E
- V
2. E
- t
- E
3. E
- S
- (
4. E
- S
- V
- E
5. E
- E
6. E
- E
- E
7. E
- (
- (
- S

process-  
d for the  
machines  
d in the  
oretical

The proposed parallel algorithm has been compared with another parallel algorithm for calculating makespan in permutation flow shop from [2]. Fig. 4 shows a speedup obtained by a parallel algorithm from [2] – FSCmax2 and a speedup obtained by a proposed algorithm – FSCmax1. Compared algorithms were tested on the test instances with a big number of jobs ( $m = 500$ ) and a different number of machines ( $m = 10, 20, 50, 100$ ). For both algorithms the speedup value increases with the number of machines. For the FSCmax1 algorithm the speedup is bigger than the speedup for FSCmax2 algorithm.

## 4 Conclusions

The method proposed in this paper can be used for computations' acceleration in metaheuristics solving the flow shop scheduling problem. The calculation time of goal function in algorithms which solve the job shop problem take even over 90% of the whole algorithm computation time, so the use of parallel algorithm for goal function calculation might result in significant decreasing of algorithm execution time for solving the flow shop problem. The theoretical results proposed here were fully confirmed by the conducted computational experiments.

## References

1. Bożejko, W.: A new class of parallel scheduling algorithms. Monographs series. Wrocław University of Technology Publishing House (2010)
2. Bożejko, W., Smutnicki, C., Uchroński, M.: Parallel Calculating of the Goal Function in Metaheuristics Using GPU. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009, Part I. LNCS, vol. 5544, pp. 1014–1023. Springer, Heidelberg (2009)
3. Bożejko, W., Wodecki, M.: Parallel Path-Relinking Method for the Flow Shop Scheduling Problem. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part I. LNCS, vol. 5101, pp. 264–273. Springer, Heidelberg (2008)
4. Bożejko, W., Wodecki, M.: Parallel Scatter Search Algorithm for the Flow Shop Sequencing Problem. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 180–188. Springer, Heidelberg (2008)
5. Bożejko, W., Pempera, J.: Parallel Tabu Search Algorithm for the Permutation Flow Shop Problem with Criterion of Minimizing Sum of Job Completion Times. In: Conference on Human System Interaction HSI 2008. IEEE Computer Society (2008) 1-4244-1543-8/08/(c)2008 IEEE
6. Bożejko, W., Wodecki, M.: Applying Multi-Moves in Parallel Genetic Algorithm for the Flow Shop Problem. In: Computation in Modern Science and Engineering: Proceedings of the International Conference on Computational Methods in Science and Engineering 2007 (ICCMSE 2007): Volume 2, Part B. AIP Conference Proceedings, vol. 963, pp. 1162–1165 (2007)
7. Bożejko, W., Wodecki, M.: On the theoretical properties of swap multimoves. Operations Research Letters 35(2), 227–231 (2007)
8. Garey, M.R., Johnson, D.S., Seti, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129 (1976)

Speedup -  
2.7998  
4.9792  
7.9685  
2.9336  
4.9642  
7.1753  
3.0548  
5.1939  
7.4627  
5.3267  
7.6555  
7.7567

9. Grabowski, J., Wodecki, M.: A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Computers & Operations Research* 31, 1891–1909 (2004)
10. Nowicki, E., Smutnicki, C.: A fast tabu search algorithm for the permutation flow shop problem. *European Journal of Operational Research* 91, 160–175 (1996)
11. Reeves, C.: Improving the Efficiency of Tabu Search for Machine Sequencing Problems. *Journal of Operational Research Society* 44(4), 375–382 (1993)
12. Taillard, E.: Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research* 47(1), 65–74 (1990)

Sc

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50